IMPLEMENTATION AND RECONFIGURATION OF ROBOT OPERATING SYSTEM ON HUMAN FOLLOWER TRANSPORTER ROBOT

Addythia Saphala
Department of Mechatronics
Faculty of Engineering
Swiss German University
Tangerang 15339, Indonesia
Email: addythia@gmail.com

Prianggada I Tanaya
Department of Industrial Engineering
Faculty of Engineering
Swiss German University
Tangerang 15339, Indonesia
Email: prianggada.itanaya@sgu.ac.id

***Abstract*—Robotic Operation System (ROS) is an important platform to develop robot applications. One area of applications is for development of a Human Follower Transporter Robot (HFTR), which can be considered as a custom mobile robot utilizing differential driver steering method and equipped with Kinect sensor. This study discusses the development of the robot navigation system by implementing Simultaneous Localization and Mapping (SLAM).**

***Keywords*: Robot Operating System (ROS); Human Follower Transporter Robot; Simultaneous Localization and Mapping; Robot Navigation.**

I. INTRODUCTION

Robotic Operation System (ROS) is important to provide a flexible framework for developing robot related applications [1]. ROS should be flexible and capable to provide distributed computations, software reuse, and rapid testing [2]. The modular nature of ROS framework enables every aspect of the system to work separately and efficiently in regards to communications between nodes. By using ROS, community can develop algorithms for various purposes that are easily integrated to the framework and preserve the works for public. The flexible nature of ROS enables for coupling and message passing between existing processes and facilitates rapid testing even for applications that require advanced and stable data stream.

One of such systems has been developed previously for Human Follower Transporter Robot (HFTR) [3, 4] (see Fig. 1). The references have developed the robot to be able to track and follow object based on the object color. However, the existing planning, execution, and monitoring (PEM) architecture is not sufficient to evade dynamic obstacles. In addition, developing a completely new algorithm requires significant amount of time. The existing program is not easily integrated to the next development phase for various reasons.

A number of studies related to ROS and HFTR has been previously conducted. Reference [5] studied the dynamic characteristics of bipedal robots. Reference [6] studied Segway robotic platform. Refer-

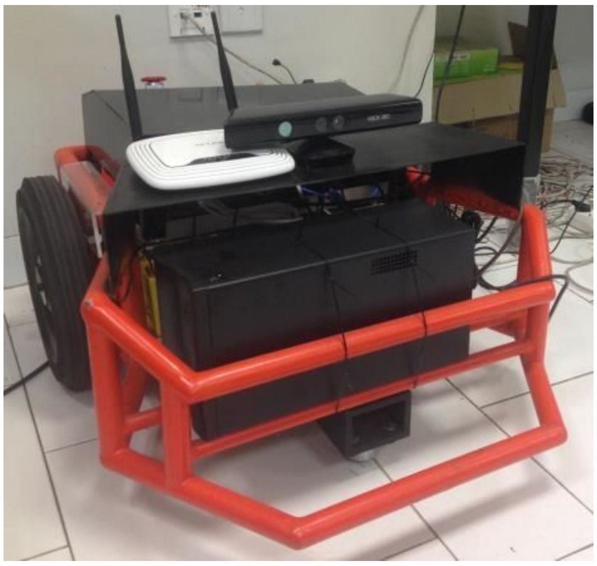


Fig. 1. An example of the Human Follower Transporter Robot (HFTR).

Received: June 10, 2015; received in revised form: July 25, 2015; accepted: July 28, 2015; available online: July 28, 2015.

ence [7] studied navigation of wheel-legged hydraulic robot. [8] proposed systems and methods for robotic transport. Reference [9] discussed a near-term autonomy of follower robots.

This study intends to develop an improvement of HFTR navigation system. The system requires several inputs to function correctly such as the robot odometry, sensor, map, and data transformation. By utilizing ROS, the existing sensor system (Kinect) can be adapted for appropriate input for mapping and navigation (laser scan). It would enable implementation of simultaneous localization and mapping (SLAM), which is required to generate maps. HFTR would also be reconfigured to use the existing ROS framework and to transform data from encoder. Reconfiguration HFTR is often unavoidable because most of the existing driver is hard to be encapsulated to the ROS framework. Sometimes, using a compatible driver is preferable instead of adapting the old one.

II. RESEARCH METHODS

To implement and reconfigure the navigation stack of ROS for HFTR, Fig. 2 shows the dependency of the stack to the other modules. The stack depends on sensor sources, map, data transform, odometry, and base controller.

Firstly, the requirements for navigation stack are sensor data, odometry data, and map. The sensor data format follows the laser scan format to conserve the computing power. The HFTR is already equipped with Kinect camera; thus, a converter is required from a depth image to laser scan. Consequently, Kinect camera driver is also required, which in turn, requiring data transformation where the data stream from their origin to the rest of HFTR.

Secondly, the navigation stack requires odometry that supplying information of the robot movement and it is handled by the Differential Driver stack in the ROS. The Differential Driver stack requires: PWM adapters, which send a velocity command for robot motors, in this case, is to Arduino board through a serial port, and robot definition for virtual simulations. The Odometry also requires Data Transform to record the robot pose.

III. RESULTS AND DISCUSSION

In this section, we present the results of the implementation and reconfiguration of the HFTR operating system. Firstly, in order to utilize the navigation stack [2], the robot has to be configured in a specific manner.

Figure 3 shows the overview of the required configuration. The components in white boxed are those that have already been implemented. Those in gray are optional components and have also been implemented. Those in blue should be created for each robot platform. The navigation stack requires a transform configuration, which is unique for each robot, sensor information in a laser scan data type, odometry information, base controller, and map [10].

The above navigation stack requires a static map, which is created by Simultaneous Localization and Mapping (SLAM) module. A gmapping package is a ROS wrapper for OpenSlam’s gmapping. This package provides ROS a node called `slam_gmapping`, which is a laser-based SLAM. Using this stack, a two-dimension occupancy grid map can be created. To utilize this package, the robot need to be able to steam a laser scan data, as well as provides a relatively accurate odometry data and transforms [11].

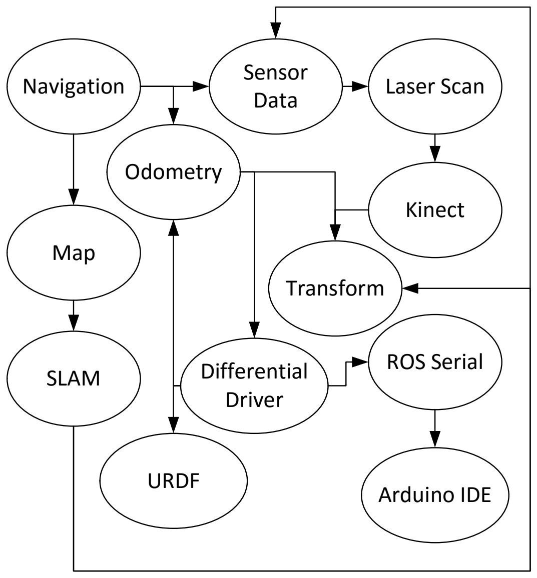


Fig. 2. The connection between modules in the robot operating system. The directed line denotes dependency, for example, module Navigation requires module Map.

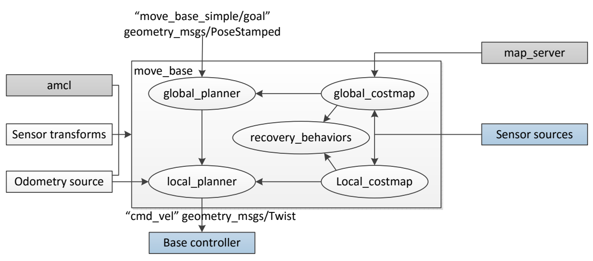


Fig. 3. The navigation stack overview [4]. Those components in white blocks have been developed, in gray are optional, and in blue are those created for each robot platform.

Cite this article as: A. Saphala and P. I. Tanaya, “Implementation and Reconfiguration of Robot Operating System on Human Follower Transporter Robot,” CommIT (Communication & Information Technology) Journal 9(2), 59–65, 2015.

In addition, the navigation stack also requires differential driver. The `differential_drive` stack provides basic tools to interface with differential drive robots with the ROS navigation stack. This stack can take twist message from navigation stack or other nodes, and publish messages for left wheel and right wheel of a differential drive robot. It also receives feedback from wheels encoder and generates transform messages as required by the ROS navigation stack [6]. This package provides four nodes [7]: `diff_tf` that provides transform for the robot base, `pid_velocity` that is a basic PID controller for motor speed, `twist_to_motors` that translates twist messages to a two-motor velocity target for the differential drive robots, and `virtual_joystick` that is a GUI controller with twist output (see Fig. 4).

Figure 5 shows active nodes and topics from `rqt_graph` for navigation stack. Figures 6–9 show the enlarged portions of Fig. 5, and the figures respectively are the left, center, upper right, and lower right sections.

The active nodes include driver sections, sensor sections, navigation sections, localization nodes, and map server. The navigation section is centered on the topic `move_base` which governing where the robot should go through twist topic. The localization node, `amcl`, functions as pose matcher for the robot. The `map_server` node simply publishes a static map for the navigation section.

The launch file for the nodes in Fig. 5 is shown in Fig. 10. To execute `map_server` and `amcl` nodes, the code needed in launch file is displayed in Fig. 10. It also displays the launch code for `move_base` node which retrieve parameters from:

- `costmap_common_params.yaml`,
- `local_costmap_params.yaml`,
- `global_costmap_params.yaml`, and

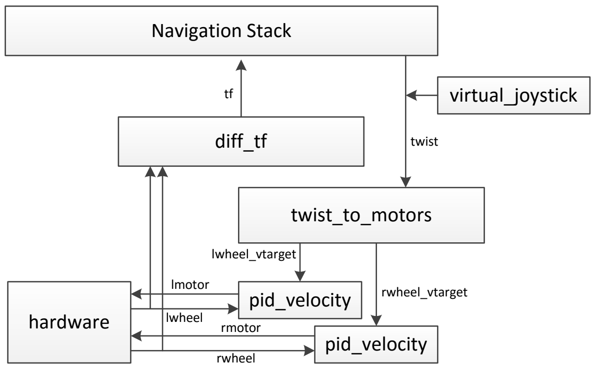


Fig. 4. The differential_drive package [12].

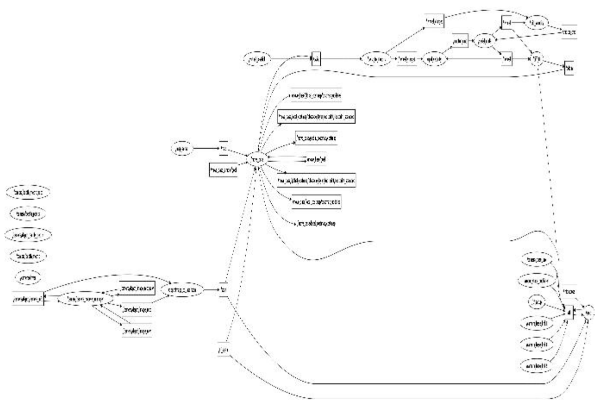


Fig. 5. The navigation nodes from `rqt_graph`.

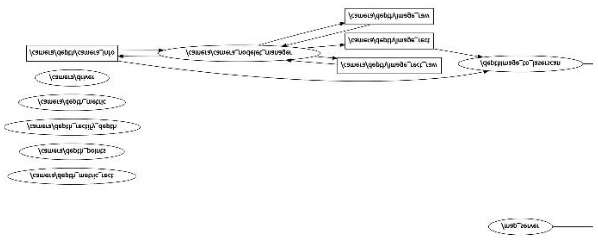


Fig. 6. The navigation nodes from `rqt_graph`: Section 1.

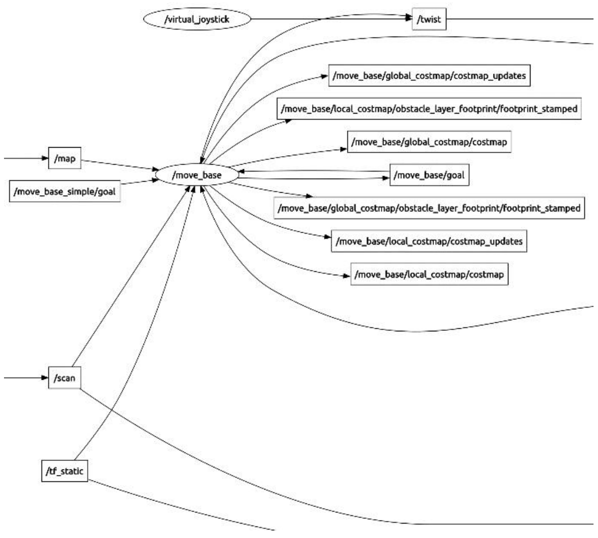

/virtual_joystick
/twist
/move_base/global_costmap/costmap_updates
/move_base/local_costmap/obstacle_layer_footprint/footprint_stamped
/move_base/global_costmap/costmap
/map
/move_base
/move_base/goal
/move_base_simple/goal
/move_base/global_costmap/obstacle_layer_footprint/footprint_stamped
/move_base/local_costmap/costmap_updates
/move_base/local_costmap/costmap
/scan
/tf_static

Fig. 7. The navigation nodes from `rqt_graph`: Section 2.

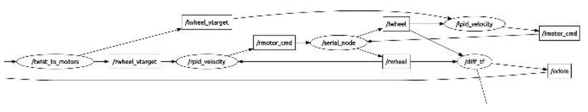


Fig. 8. The navigation nodes from `rqt_graph`: Section 3.

- `base_local_planner_params.yaml`.

In Fig. 11, the `costmap` common parameters is presented. These parameters are used by both global `costmap` and local `costmap`. In this file, the obstacle range and ray trace range is defined, which means at which range the obstacle is detected and at which range the robot would attempt to seek a clear path. The footprint is the shape of the robot on two dimension, which have to be defined since the HFTR cannot represented by a circle but by a polygon. The observation source is the sensor input, which in this case is laser scan sensor.

Figure 12 shows the local `costmap` parameters. It simply defines the publish rate and update rate of the map, and local costmap's properties. It also defines the odometry topic and robot base.

Figure 13 shows the global `costmap` parameters, which define map topic, update frequency and robot base. Figure 14 displays the base local planner parameters, in this file, the maximum and minimum velocity and acceleration both translation and rotation of the robot is defined. Figure 15 shows a virtual representation of the HFTR with a static map inside rviz. The map used is that of FB311. In Fig. 16 the global costmap is shown on top of static map. In Fig. 17 the local costmap is shown on top of the static map. Finally, in Fig. 18 both map are shown on top of the static map. In Fig. 19, sending a navigational goal is shown, and in Figure 20, the virtual representation of HFTR can be moved inside rviz by sending two-dimension position estimation data [13].

We identified a few limitations on the current development and they may become topics for future

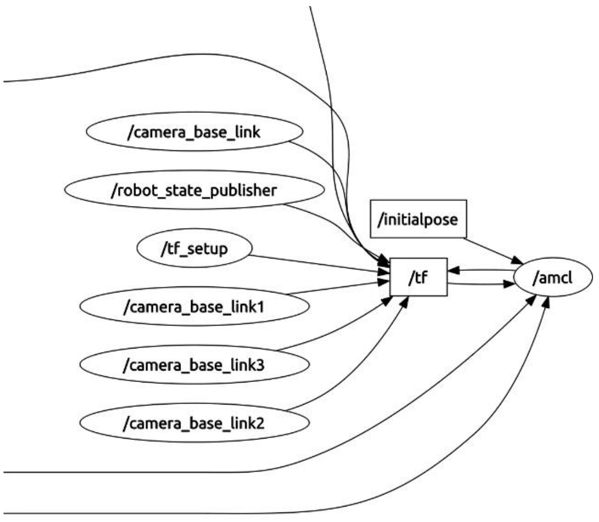


Fig. 9. The navigation nodes from `rqt_graph`: Section 4.

```
<launch>
  <node pkg="rosserial_python" type="serial_node.py" name="serial_node">
     <rosparam>
        port: /dev/ttyACM0
     </rosparam>
  </node>
  <param name="robot_description" textfile=˜$(find teegut)/
     urdf/teegut.urdf"/>
  <rosparam param="ticks_meter">694,5</rosparam>
  <node pkg="differential_drive" type="pid_velocity.py"
     name="lpid_velocity">
     <remap from="wheel" to="lwheel"/>
     <remap from="motor_cmd" to="lmotor_cmd"/>
     <remap from="wheel_vtarget" to="lwheel_vtarget"/>
     <remap from="wheel_vel" to="lwheel_vel"/>
     <rosparam param="Kp">276</rosparam>
     <rosparam param="Kt">2160</rosparam>
     <rosparam param="Kd">8.5</rosparam>
     <rosparam param="out_min">255</rosparam>
     <rosparam param="out_max">255</rosparam>
     <rosparam param="rate">30</rosparam>
  </node>
  <node pkg="differential_drive" type="pid_velocity.py"
     name="rpid_velocity">
     <remap from="wheel" to="rwheel"/>
     <remap from="motor_cmd" to="rmotor_cmd"/>
     <remap from="wheel_vtarget" to="rwheel_vtarget"/>
     <remap from="wheel_vel" to="rwheel_vel"/>
     <rosparam param="Kp">276</rosparam>
     <rosparam param="Kt">2160</rosparam>
     <rosparam param="Kd">8.5</rosparam>
     <rosparam param="out_min">255</rosparam>
     <rosparam param="out_max">255</rosparam>
     <rosparam param="rate">30</rosparam>
  </node>

  <node pkg="differential_drive" type="virtual_joystick.py"
     name="virtual_joystick" output="screen"/>

  <node name="robot_state_publisher" pkg="robot_state_publisher"
     type="state_publisher"/>

  <node name="tf_setup" pkg="transformf_setup" type="transform" />

  <include file="$(find freenect_launch)/launch/examples/
     freenect-xyz.launch"/>

  <node pkg="depthimage_to_laserscan" type=depthimage_to_laserscane"
     name"depthimage_to_laserscane">
     <remap from="image" to="camera/depth/image_rect" />
     <remap from="camera_info" to="camera/depth/camera_info" />
  </node>

  <!--- Run the map server -->
  <node name="map_server" pkg="map_server" type="map_server"
     args="$(find maps)/map.pgm 0.05"/>

  <!--- Run AMCL -->
  <include file="$(find amcl)/examples/amcs_diff.launch"/>

  <node pkg="move_base" type="move_base" respawn="false"
     name="move_base" output="screen">
     <rosparam file="$(find teegut)/costmap_commom_params.yaml"
     command="load" ns="global_costmap"/>
     <rosparam file="$(find teegut)/costmap_common_params.yaml"
     command="load" ns="local_costmap"/>
     <rosparam file="$(find teegut)/local_costmap_params.yaml"
     command="load"/>
     <rosparam file="$(find teegut)/global_costmap_params.yaml"
     command="load"/>
     <rosparam file="$(find teegut)/base_local_planner_params.yaml"
     command="load"/>
     <remap from="cmd_vel" to="twist"\>
  </node>
  <node pkg="rvtz" type="rviz" output="screen"/>
</launch>
```

Fig. 10. The navigation launch file.

```
obstacle_range: 1.5
raytrace_range: 2.0
footprint: [[-0.19,0.24],[-0.19,-0.24],[0.19,-0.24],[0.328564,-0.32],
   [0.553564,-0.19],[0.553564,0.19], [0.328564,0.32], [0.19,0.24]]
#robot_radius: ir_of_robot
inflation_radius: 0.25

observation_sources: laser_scan_sensor

laser_scan_sensor: {sensor_frame: camera_link, data_type: LaserScan,
  topic: scan, marking: true, clearing: true}

point_cloud_sensor: {sensor_frame: frame_name, data_type: PointCloud,
  topic: topic_name, marking: true, clearing: true
```

Fig. 11. `costmap_common_params.yaml`

developments. The first is regarding the processing power. The current processor is unable to process three dimensions data at an acceptable rate. Thus, a better processor can be used, or once ROS is able to run across network, the computer can be used to host nodes with high processing power requirement. In PEM architecture, the processing parts, which require high processing powers, can be placed in an appropriate computer, while the execution and monitoring can be handled in on-board mini-PC.

The second is regarding the mapping. The ideal way to create a map is to start with searching the most optimum parameters from a set of records and then use the parameters in real-time mapping. The mapping process should be done indoors within Kinect sensor range and with sufficient lighting. The mapping process may be affected by surrounding noise, this should be further studied.

```
local_costmap:
  global_frame: odom
  robot_base_frame: base_link
  update_frequency: 5.0
  publish_frequency: 2.0
  static_map: false
  rolling_window: true
  width: 6.0
  height: 6.0
  resolution: 0.05
```

Fig. 12. `local_costmap_params.yaml`

```
global_costmap:
  global_frame: /map
  robot_base_frame: base_link
  update_frequency: 5.0
  static_map: true
```

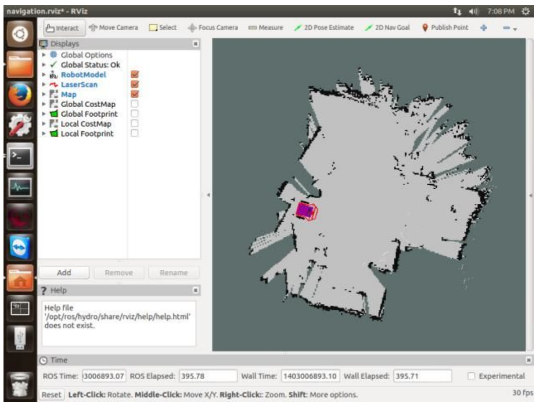
Fig. 13. `global_costmap_params.yaml`

```
TrajectoryPlannerROS:
   max_vel_x: 0.2
   min_vel_x: 0.05
   max_rotational_vel: 0.2
   min_in_place_rotational_vel: 0.05

   acc_lim_th: 0.05
   acc_lim_x: 0.05
   acc_lim_y: 0.05

   holonomic_robot: false
```

Fig. 14. `base_local_planner_params.yaml`

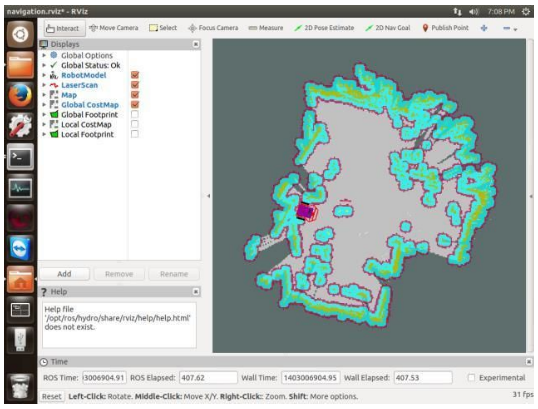


Help file '/opt/ros/hydro/share/rviz/help/help.html' does not exist.

ROS Time: 1403006893.07 ROS Elapsed: 395.78 Wall Time: 1403006893.10 Wall Elapsed: 395.71 Experimental

Left-Click: Rotate. Middle-Click: Move X/Y. Right-Click: Zoom. Shift: More options. 30 fps

Fig. 15. Virtual Representation of HFTR with Static map

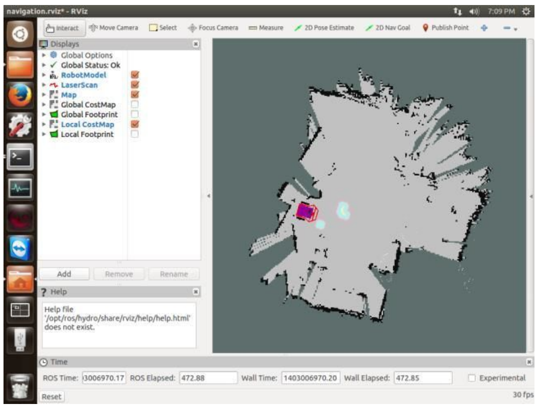


Help file '/opt/ros/hydro/share/rviz/help/help.html' does not exist.

ROS Time: 1403006904.91 ROS Elapsed: 407.62 Wall Time: 1403006904.95 Wall Elapsed: 407.53 Experimental

Left-Click: Rotate. Middle-Click: Move X/Y. Right-Click: Zoom. Shift: More options. 31 fps

Fig. 16. Virtual Representation of HFTR with Global Costmap

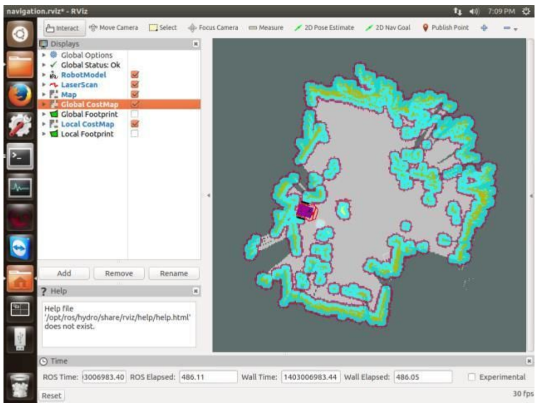


Help file '/opt/ros/hydro/share/rviz/help/help.html' does not exist.

ROS Time: 1403006970.17 ROS Elapsed: 472.88 Wall Time: 1403006970.20 Wall Elapsed: 472.85 Experimental

30 fps

Fig. 17. Virtual Representation of HFTR with Local Costmap

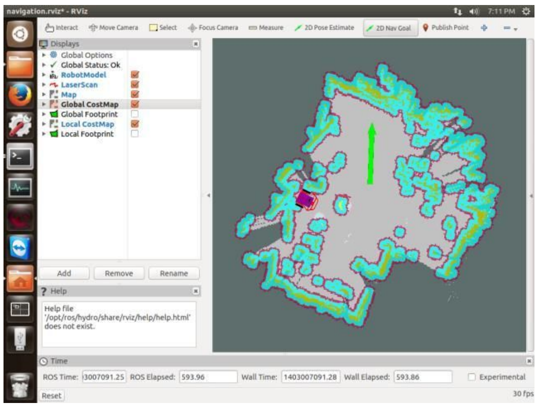


Help file '/opt/ros/hydro/share/rviz/help/help.html' does not exist.

ROS Time: 1403006983.40 ROS Elapsed: 486.11 Wall Time: 1403006983.44 Wall Elapsed: 486.05 Experimental

30 fps

Fig. 18. Virtual Representation of HFTR with Global and Local Costmap

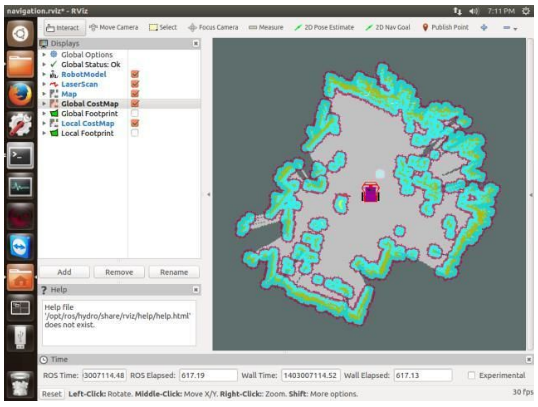


Help file '/opt/ros/hydro/share/rviz/help/help.html' does not exist.

ROS Time: 3007091.25 ROS Elapsed: 593.96 Wall Time: 1403007091.28 Wall Elapsed: 593.86 Experimental

30 fps

Fig. 19. Sending Navigation Goals through rviz



Help file '/opt/ros/hydro/share/rviz/help/help.html' does not exist.

ROS Time: 3007114.48 ROS Elapsed: 617.19 Wall Time: 1403007114.52 Wall Elapsed: 617.13 Experimental

Left-Click: Rotate. **Middle-Click:** Move X/Y. **Right-Click:**: Zoom. **Shift**: More options.

30 fps

Fig. 20. Moving Virtual Representation of HFTR in rviz

The last is regarding the navigation. It is important to perform experiments with static and dynamic obstacle avoidance using navigation stack. Currently, only holonomic and differential drive mobile robots is supported by navigation stack. A system or package should be developed for other types of robots with same or similar function of navigation stack.

IV. Conclusions

The Robot Operating System software has been implemented and reconfigured to work with Human Follower Transporter Robot. The following are the related packages. The ROS serial package is required to enable communication through serial port with Arduino, which controls DC motors. The `differential_drive` package can act as a motors driver for the HFTR, as well as providing odometry and transform data. Kinect sensor can be utilized with ROS software; there are drivers for Kinect available within ROS framework, `freenect_launch` package, which include with coordinate frame transform program associated with Kinect's shape. ROS framework structure lends a flexibility to change the sensor data input, it enable streaming two dimension laser scan data from Kinect sensor. The `slam_gmapping` node works well with HFTR, the generated map can be used directly for navigation purpose without external changes. Navigation stack is able to be implemented and reconfigured for HFTR. It can control the movement of the HFTR reasonably well. Despite the unique mechanical design of the HFTR, ROS is able to function well with it. Thus it can be concluded that: ROS framework is generic, which means it can work with a lot of robot types. The absence of HFTR from ROS database and library means that the developed programs can be made into new package for HFTR in ROS library. Robot model which can be used to virtually represent the HFTR can be used in rviz. The models evolution also can be seen as implementation process of ROS. With better implementation of ROS, various parameters and data can be included in URDF file for more complete description, which results in more detailed model. From the navigation experiments, it can be concluded that: the navigation stack can be implemented and reconfigured in HFTR; the navigation stack can register static map and obstacle; the occupational value in grid map can be seen from global costmap (wall) and local costmap (obstacle) which depends on distance from the object (inflation); and, the navigation pose and goal can be sent through rviz.

References

[1] J. M. O'Kane. (2014) A gentle introduction to ros. Retrieved on January 2015. [Online]. Available: http://www.cse.sc.edu/~jokane/agitr/

[2] E. M. Eppstein. (2014) Navigation. Retrieved on January 2015. [Online]. Available: http://wiki.ros.org/navigation.

[3] F. K. Mista, “Development of trajectory planner based on lagrange polynomial and b-spline equations for an autonomous human follower transporter robot,” Master's thesis, Department of Mechatronics, Faculty of Engineering, Swiss German University, 2013.

[4] W. Tjiu, “Development of execution and monitoring architecture modules for an autonomous human follower transporter robot,” Bachelor Thesis, Department of Mechatronic, Faculty of Engineering and Information Technology, Swiss German University, Tangerang, Indonesia, 2013.

[5] Nirmala, P. I. Tanaya, and M. Sinaga, “A study on bipedal and mobile robot behavior through modeling and simulation,” *International journal*

of communication and information technology, vol. 9, pp. 1–10, 2015.

[6] H. G. Nguyen, J. Morrell, K. D. Mullens, A. B. Burmeister, S. Miles, N. Farrington, K. M. Thomas, and D. W. Gage, “Segway robotic mobility platform,” in *Optics East*. International Society for Optics and Photonics, 2004, pp. 207–220.

[7] C. Y. Wong, K. Turker, I. Sharf, and B. Beckman, “Posture reconfiguration and navigation maneuvers on a wheel-legged hydraulic robot,” in *Field and Service Robotics*. Springer, 2015, pp. 215–228.

[8] J. Zhang, A. Salerno, N. Simaan, Y. L. Yao, G. Randers-Pehrson, G. Garty, A. Dutta, and D. J. Brenner, “Systems and methods for robotic transport,” Patent, Aug. 31, 2010, uS Patent 7,787,681.

[9] B. E. Brendle Jr and J. J. Jaczkowski, “Robotic follower: near-term autonomy for future combat systems,” in *AeroSense 2002*. International Society for Optics and Photonics, 2002, pp. 112–117.

[10] Wiki. (2014) Navigation/tutorials/robotsetup. Retrieved on January 2015. [Online]. Available: http://wiki.ros.org/navigation/Tutorials/RobotSetup

[11] B. Gerkey. (2014) gmapping. Retrieved on January 2015. [Online]. Available: http://wiki.ros.org/gmapping

[12] J. Stephan. (2014) differential_drive. [Online]. Available: http://wiki.ros.org/differential_drive

[13] A. Saphala, “Reconfiguration and implementation of robot operating system for mapping and navigation on human follower transporter robot,” Master Thesis, Department of Mechatronic, Faculty of Engineering and Information Technology, Swiss German University, Tangerang, Indonesia, 2014.